ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(10), DOI: 10.17485/ijst/2016/v9i10/88893, March 2016* ISSN (Online) : 0974-5645

Generation of Variant Random Order (VRO) in Text Graphics Color CAPTCHA for Enhancing Web Security Protection

S. PradeepKumar[1*], R. Ramachandaran[2] and A. Saravanan[3]

[1]Department of CSE, Sathyabama University, Chennai - 600119, Tamil Nadu, India; pradeepk.feb@gmail.com
[2]ECE, Dhanalakshmi College of Engineering, Chennai - 601301, Tamil Nadu, India; chandra557@yahoo.co.in
[3]CSE, Rajalakshmi Institute of Technology, Chennai - 600124, Tamil Nadu, India; saromanj19@gmail.com

Abstract

Objectives: In the modern era of web services, the web site registration process plays a vital role for authorized access. The process can be easily cracked by the hackers through their false registration. To overcome this problem, we introduce a Variant Random Order (VRO) in text-graphics color CAPTCHA approach for enhancing web security. **Methods:** In our proposed approach, the user has to enter all the characters based on the random number generation model, where the random number is to be displayed under the corresponding CAPTCHA characters. Based on the sorted order (generally ascending) of the random number, the user has to select the CAPTCHA characters one by one and to fill in the given empty text box. After typing all the characters from a given CAPTCHA, then the user can sign and enter into the required web area. If any mistake made in the entry process, a new CAPTCHA will be generated for the same login process. Consequently, we say that the process will be unbreakable by any Robot and in turn the website access is completely restricted. The number of CAPTCHA characters along with a random number is to be varied for every attempt. Also, each and every character in the CAPTCHA is to be represented with different colors, which in turn the pixel intensity can be varied for the same character representation. **Findings: Applications:** This type of techniques helps the proposed system to display the CAPTCHA with additional security in major application areas such as health services, military services and banking services.

Keywords: CAPTCHA-Completely Automated Public Turing Test to Tell Computer and Human Apart, Text Graphics Color, VRO-Variant Random Order, Web Access, Web Security

1. Introduction

CAPTCHA is a program and it is used to perform test for the user and to enter into the web site area. The security of a CAPTCHA is that no other computer program cannot pass the test even if the knowledge of a working CAPTCHA is known. Such type of test can be referred as 'challenge response' test in which the user has to give the correct answer either through human feed or a 'bot' (computer automated software program). At present, hackers utilize a modern tool for having the chance to break a CAPTCHA test through 'false registration'.

Hence, this proposed work introduces a variant random order CAPTCHA in order to display a CAPTCHA over the webpage. All the necessary information are filled by the user after that only the CAPTCHA page will be appeared. The representation of CAPTCHA must be a human friendly and it can restrict the characters that slightly be collided with each other. So, the user may not irritate at any point of time while entering the CAPTCHA characters. In this case, the CAPTCHA is refreshed at a maximum of four times until the user submits the final CAPTCHA. Some of the drawbacks of the existing techniques are discussed below.

Hence, this proposed work introduces a variant random order CAPTCHA in order to display a CAPTCHA

* *Author for correspondence*

over the webpage. All the necessary information are filled by the user after that only the CAPTCHA page will be appeared. The representation of CAPTCHA must be a human friendly and it can restrict the characters that slightly be collided with each other. So, the user may not irritate at any point of time while entering the CAPTCHA characters. In this case, the CAPTCHA is refreshed at a maximum of four times until the user submits the final CAPTCHA.

The Figure 1 shows some of the Google CAPTCHA characters. From this type of CAPTCHAs, the user may be confused due to the collision of characters and similar colors used in the CAPTCHA representation[1].

In general, the following three major types of CAPTCHA are discussed by many researchers on their research works[2]. Such types of CAPTCHA are:

1.1 Text based CAPTCHA's

Text based CAPTCHA normally consists of an image with 5 to 8 characters in it. This image will have some noise due to the horizontal or cross lines are cracked in the CAPTCHA characters[3]. Let us consider the examples given in the EZ-Gimpy and Baffle Text[4], where the various forms of disturbances/noises are occurred. Many websites use the different types of their own CAPTCHAs representation (Figure 2).

1.2 Image based CAPTCHA's

Image based CAPTCHA is a graphics based CAPTCHA in which the user have to find out similarity between the two set of images in a various block. In the case of Bongo CAPTCHA[5], the two sets of images are displayed in a single block, where the user has to find out the distinct characters among them (Figure 3). This type of applications is to be focused in a visual recognition problem discussed by Baljit Singh Saini[5].

1.3 Audio based CAPTCHA's

It is a sound based system and it contains downloadable audio clips. This CAPTCHA mainly developed for visually disabled users. In this type of applications, the user has to put much more efforts to recognize a spoken word.

In general, good CAPTCHAs must satisfy the following characteristics:

- It should be a user friendly.
- The tester machine can be easily to generate the CAPTCHA.
- Bots cannot have any chance to predict the CAPTCHA characters.

Some of the CAPTCHA based applications are: Protection in web registration, Preventing in online polls, Preventing in dictionary attacks, Preventing – Emails and Online games[6].

Figure 1. Confusing character in a google CAPTCHA.

Figure 2. Microsoft CAPTCHA Yahoo CAPTCHA.

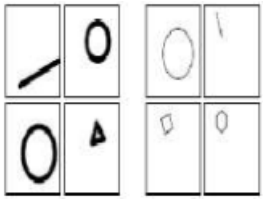

Figure 3. Bongo CAPTCHA Example.

In this paper, the section 2 describes the new proposed method for utilizing the variant random order CAPTCHAs. The section 3 gives the idea of breaking character CAPTCHA scheme. The section 4 shows the implementation of variant random order in text graphics character CAPTCHA scheme. Then, the section 5 discusses the analysis of CAPTCHAs report performed by the user and the efficiency is reported. Finally, the section 6 describes the conclusion and future enhancement on CAPTCHA scheme.

2. Proposed Work

The security of displaying CAPTCHA should be very strong and even the bots could be confused to break a CAPTCHA. In general, the entire web applications' user has to fill all the CAPTCHA characters as per the given form. But, the work proposed in this paper, the

user has to fill all the CAPTCHA characters based on the order prescribed by random number generation for enhancing the security purpose. So, we introduce a Variant Random Order CAPTCHA that can be used to generate a distorted test along with random numbers. In such a case, a separate number is to be displayed along with each CAPTCHA character. Then the user has to fill all the characters based on the sorted number given in the display. The random number display is to be differed in various ways by using the same number of CAPTCHA character or same CAPTCHA representation. For an every attempt, the number of CAPTCHAs characters is to be varied. Each and every character is to be represented in different color. The same character can have the chance to represent different colors. So, the number of pixels used and intensity value are to be varied for the same character representation. So the hackers cannot have the chance to predict a CAPTCHA character representation through this proposed methodology.

3. Breaking CAPTCHA's Character Scheme

The sample CAPTCHA character is shown in the Figure 4, which has been distorted and tilted in different positions. In general, the alphabetic characters (a to z: upper and lower case) and numeric symbols (0,1.. 9) are used in the CAPTCHA representation. Hence, the representation and display of CAPTCHA may have several combinations to ensure the securable web access. (Figure 4)

3.1 Working Principles

Each and every CAPTCHA character is to be represented in different colors. The random sorted characters are used to represent a word, which is very difficult to predict and guess.

- Create random value: For an image, the number of random values used is equal to the number of CAPTCHA characters. Each and every character is displayed with corresponding random value.
- User entry: Based on the sorted random order number generation, user has to fill up all the CAPTCHA characters in the given text box.

Two components are used in the CAPTCHA character representation. One component acts as a primary and another one may act as a secondary. The primary component consists of 'white color' which refers to the background text and the secondary component refers to the CAPTCHA character that has to be represented in different colors. The black color is used to crack the CAPTCHA character. The dynamic order is to be varied w.r.t to the number of CAPTCHA character used in the particular 'testing' process (Figure 5).

3.2 Preprocessing

Before going to the segmentation, security of text based CAPTCHA can be analyzed. The CAPTCHA should hold constraints for doing preprocessing.

(i) Readable - All the CAPTCHA characters are readable and it should be apparent only to the human user[7]. Cracks are represented by line, dots and curves in the CAPTCHA scheme. User should not get irritated at any point of time.

(ii) Stability - Due to the various colors used in CAPTCHA representation, even if the Bots (even if sophisticated software used) cannot have the chance to detect the CAPTCHA character through many attempts. Each and every attempt, the same character can have the chance to represent different colors.

(iii) Encoding mechanism - Noise removal - Sometimes, the representation of CAPTCHA with noise (crack) can improve the strength of CAPTCHA[8]. The black color can be used to crack the entire CAPTCHA image scheme. For removing the black color, we can scan the entire image for making the verification process at pixel level. After getting a black pixel, we can check the surrounding/neighboring eight pixels[9].

The following methods (Figure 6) are used for smoothing the CAPTCHA image.

(iv) Image Smoothing - The above process is to remove noise that could harm later steps involved in the preprocessing procedure[9]. Now the CAPTCHA image contains many colors and the hackers cannot have any chance to identify the characters[3]. Hence, it is very difficult to work on 256 intensity values for identifying CAPTCHA at particular time interval. In this case, the color of the image contains 'white' which refers the background color and the rest of the colors (different color) are used to refer the foreground color.

3.3 Segmentation Process

Each and every CAPTCHA character can have own pixel count. Based on the usage of the colors, the number of

pixels can be varied for the same character representation. The characters are to be distorted in different shape and tilted in some angle[10]. The following snake segmentation algorithm is used to detect the crack (black pixel) in the CAPTCHA screen. The speed of the snake segmentation algorithm depends upon the cracks and number of characters in the CAPTCHA set. Each and every character can have the chance to represent different colors. The look up table (Table 1) shows the number of pixel count varies with the same character representation.

Algorithm: *Snake Segmentation*

Step 1: Start with bottom left pixel coordinate value,
P(i,j) → (1,1)
/*usually foreground color is white*/
/* i&j defines the x-coordinate and y-coordinate */
Begin

Step 2: {do the process for the entire CAPTCHA screen}
If(j<=h)(h:text_height)

Step 3: {Track the same pixel value towards up, neglect all the black color pixel value until it reaches text_height
P(i,j) →(1,h)(h:text_height)

Step 4: Turn right one pixel position,
P(i,j)→ ($x_{i+1,h}$)
Again track the same pixel value towards down

Step 5: Bottom height pixel coordinate value is
P(i,j) → P($x_{i+1,1}$)

Step 6: Turn right one pixel position,
P(i,j) → P($x_{i+2,1}$)
{Repeat steps 3 to 5 until to find a new pixel value}

Step 7: If new pixel value obtain, vertical slicing do→ to break a segment in a text

Step 8: All the above steps 3 to 7 are to be repeated until separation of all characters in a screen text
End

Advantages:

i. Character segmentation is done well.

ii. Bots (using sophisticated software) cannot have the chance to detect the CAPTCHA's characters

iii. Conflict to many attacks during the time of preprocessing, pixel count and dictionary attack[11].

Figure 4. Shows a different color character CAPTCHA.

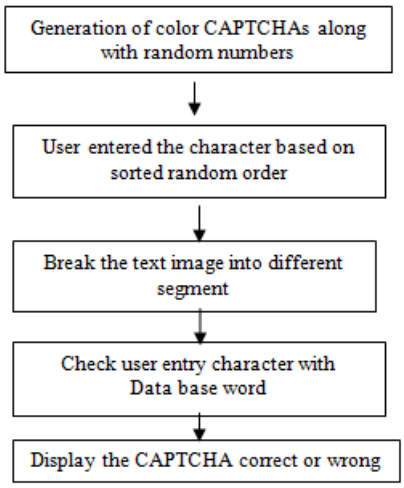

Figure 5. Working principles of a CAPTCHA.

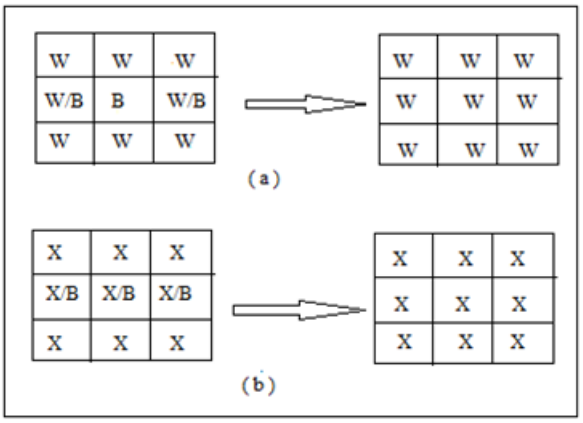

Figure 6. Converting a pixel gap (B- represents black color pixel, W- represents white color pixel, X- represents cyan, magenta, red, green, blue, brown color pixels).

Table 1. Lookup table entries

Letter	Colorused	Pixelcount
e	Brown	168
e	Green	186
e	Red	172

4. Implementation

The strength of a CAPTCHA is a function of varying length, color and different random number generation. The total number of CAPTCHA character is displayed on a screen is equal to the number of random numbers generated. Each and every character below, the corresponding random number are displayed.

4.1 Character Extraction

Once the CAPTCHA character set is completely segmented, then we can count the pixel count of individual CAPTCHA character easily[12]. The vertical segmentation algorithm should not fail at any case, even if we use the possible combination of all characters ({A,B…Y,Z}, {a,b,..y,z}{0,1,..9}) along with graphical operations (Figure 7).

The database maintains a CAPTCHA character set, labels, total pixel count of the CAPTCHA character and a recognized word pixel count which defines the sum of the individual CAPTCHA character[12]. A database can have nearly 2000 sets of CAPTCHA characters. The recognized word is not available in the English dictionary due to the combination of numeric with letters. So, this process cannot lead to the online guessing attacks, online attacks and dictionary attacks for the hackers[11].

4.2 User Entry

If user fills all the application details, then only the CAPTCHA 'testing' will be appeared. Generally, the user fills all the CAPTCHA character in the given Text box appeared above the CAPTCHA character. But, in this process user wants to fill all the characters based on the random number sorted order (Label). This makes the CAPTCHA code will be strengthened and the security is to be enhanced[13]. The label indicates the random number generator i.e., each and every CAPTCHA character appeared with corresponding label display. The total number of CAPTCHA character is equal to the number of random values displayed. The generated numbers are used in the label can have a permuted choice, even if the same number of CAPTCHA character used for testing (Figure 8).

Algorithm: *User Entry*
Begin
Step 1: CAPTCHA Character → displayed
{each character are to be in displayed
in different colors}
Step 2: User wants to submit → all character
Step 3: Labels indicate random numbers
Step 4: Number of CAPTCHA character used ≠ Number of Random Numbers displayed
Step 5: Below each CAPTCHA's character → Random value display
Step 6: Based on sorted random value
If (user entry = EMPTY)
Display a Message 'No Data:'
/* goes to beginning of a loop*/
else if (user entry = Matched)
Begin
Display a Message, 'CAPTCHA is correct'
break; /*Successful operation*/
End
else Display a Message, 'Entry is invalid'
/* it goes to beginning of a loop*/
{Step 6 is evaluated for a maximum of four times}
End

This algorithm is evaluated for the maximum of four times (i.e. wrong entry as given by the user) inorder to avoid the hackers to access information from the web. If the user exceeds the maximum attempt, then a computer system refreshes the entire web application. In this situation, once again the information has to be filled from the beginning of the work itself.

4.3 Random Number Generation (RNG)

The various statistical tests of randomness have to be carried out for CAPTCHA character set. These tests use a cryptographic hash function and to represent the CAPTCHA in securable manner. RNG provides a various combination of all available characters {(A,B,..Y ,Z),(a,b,..y,z),(0,1,..8,9)} for a given input (i.e. number of CAPTCHAs characters used). If the hackers see a lot of random number generated by the system, he/she can't predict or guess about the output.

Algorithm: *RNG*
Input: (n,a[n],key)
Output: random_Data(key,a[n])
/* n→no of characters used */
/* key→hash key function */
/* An array a[n]→ containing data set */
Set uses data{(A,B,..Y,Z),(a,b,..y,z),(0,1,..8,9)}
Key'→F(key,a[n])
/*Key'→Modified Random function*/
/* F()→Cryptographic function */
Return random data

RNG produces a much more random number choice for a given input. As much as possible, a large number of outputs produced for the given input. In such a case, if the characters form the English word (available in the dictionary) that should be avoided for representing new CAPTCHAs. In this connection, the system has maintained the database of RNG along with equal number of character and a CAPTCHA set.

4.4 Validation

User enters all the CAPTCHA characters and to check with the database. The database maintains a recognized word and sum of all the pixels count for the each CAPTCHA character set to ensure the validation process as shown in the Figure 7. The Figure 9 shows, user has to fill all the CAPTCHA's character based on a sorted label position. In such a case, the enhanced version of segmentation algorithm supporting the validation process smoothly and check the pixel count with the database. In older version, the system does not give the correct result due to the usage of same color in the character set.

Based on the scheduling algorithm, the different types of CAPTCHA character set can be appeared then and there along with a random number generator (RNG). The number of characters used in a CAPTCHA set may vary from 6 to 8.

We explore the same in the webpage display and then start the user registration process. If the user not able to submit the CAPTCHA character based on the sorted order as per the given RNG, a new set of CAPTCHA character is to be generated after a predefined time period (Figure 9).

Advantages

i. Human user can easily predict the CAPTCHA character.

ii. Segmentation can be done easily, even though the character may collide with each other.

iii. The number of used CAPTCHA characters is not fixed.(it may vary from 6 to 8)

iv. The same character can have the chance to represent in different colors.

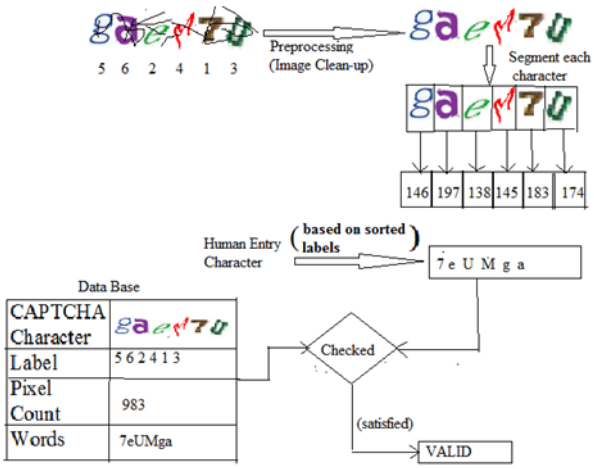

Figure 7. CAPTCHA character entry-validation by a user.

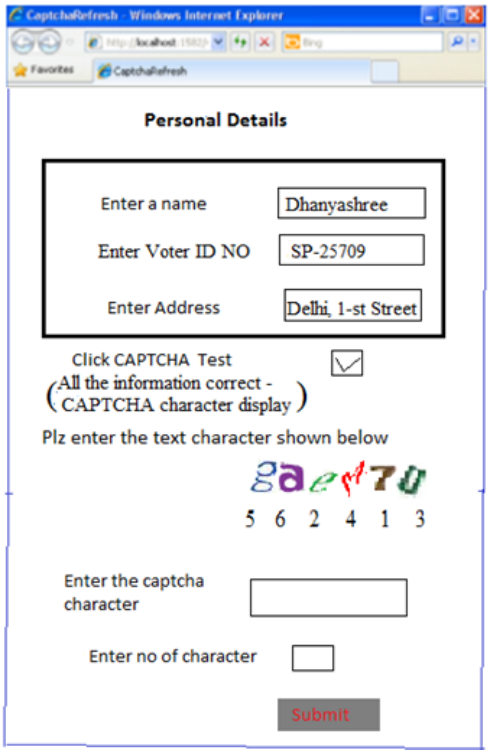

Figure 8. User Entry CAPTCHA form.

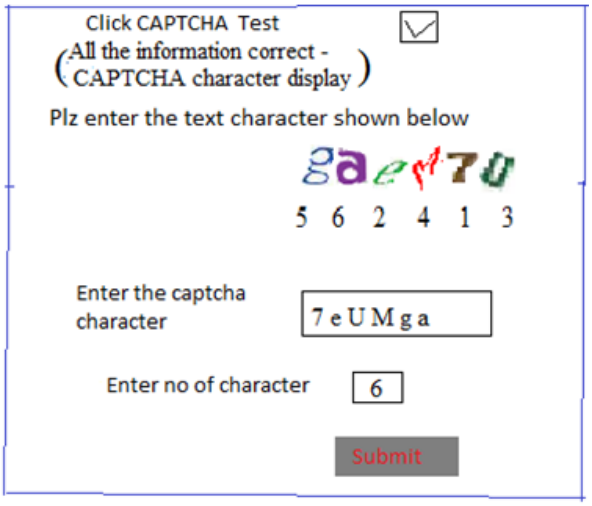

Figure 9. User Entry-CAPTCHA based on RNG.

5. Results and Discussion

The snake segmentation algorithm has been experimented successfully for calculating the pixel count of every character in a CAPTCHA, where only the black color is used to crack the entire CAPTCHA set. The segmentation process is completely depending upon the character in a CAPTCHA set. Usually, the letter 'U' and numeric '8' have valley shape and takes longer time to detect it. The success rate on a sample test has been boosted to100%.

The pixel count of every image character was not fixed due to the usage of various colors, which represents the same word. The performance evaluation is based on evaluating effectiveness of CAPTCHAS test by measuring its accuracy. The accuracy is the fraction of CAPTCHAs that were answered correctly by the human user.

The attack effectiveness is evaluated by metrics called as Coverage and Precision discussed in the following calculation[14]. In this proposed work, metric is used for finding the CAPTCHA efficiency (use coverage and precision).

Coverage metric = user	Number of CAPTCHA test/ attempt to answer
Precision metric = has to	Fraction of CAPTCHAs that answer correctly based on RNG

5.1 Accuracy

An accuracy is defined as the number of successful user entry CAPTCH's character set.

Algorithm: Accuracy_CAPTCHA_Entry

```
Begin
Var:a[]→ CAPTCHA's character in a set
        b[]→Alloted CAPTCHA character by dynamic random
number generator(RNG)
n→Number of CAPTCHA's character and also equal to
          number of RNG
 m→Number of times CAPTCHA's test conducted.
     s[]→Find user entry CAPTCHA's character
{0→failure or 1→success}
    sum→number of all successful CAPTCHA entry
x→User predict CAPTCHA's character
{guess →correct or wrong}
y→User enter CAPTCHA's character
{enter →correct or wrong}
 For i=1 to n do, i++
   Begin
     For j=1 to n do, j++
      Begin
      If(a[j]!= Human user predict CAPTCHA's character)
          x=0; break;
        Else
          x=1;
      End
     For k=1 to n do, k++
      Begin
      If(b[k]!= Human user entered CAPTCHA's character
                based on RNG)
         y=0; break;
       Else
         y=1;
      End
    If(x==1) && (y==1)
      s[i]=1; sum=sum+s[i];
    Else
      s[i]=0;
   End
End
```

The 'sum' holds a number of successful CAPTCHA entry performed by the user.

$$\text{Accuracy}=\frac{\text{sum(Number of successful CAPTCHA entry)}}{\text{N(Number of test conducted)}}$$

5.2 Analysis

The accuracy is calculated as follows:

$$Accuracy -> g(x) = \sum_{i=1}^{i=m} * \left[\sum_{j=1}^{j=n} a[j] + \sum_{k=1}^{k=n} b[k] \right] s[i] j,k$$

Where j iteration→ human user recognize 'n' number of CAPTCHA's character

k iteration→human user predicts the random number generator (RNG) i.e based on sorted RNG, user enters the CAPTCHA's character

i iteration→m times ,number of sample test conducted

a[j]=1{if the user predicts all CAPTCHA's character correctly, otherwise a[j]=0}

b[k]=1{user entered CAPTCHA's character correctly otherwise a[j]=0}

$$g(x) = \sum_{i=1}^{i=m} * \left[\sum_{j=1}^{j=n} a[j] + \sum_{k=1}^{k=n} b[k] \right] s[i]$$

$$g(x) = \sum_{i=1}^{i=m} * \left[\sum_{j=1}^{j=n} (1) + \sum_{k=1}^{k=n} (1) \right] s[i]$$

$$g(x) = \sum_{i=1}^{i=m} * [(n-1+1)+(n-1+1)](1)$$

Finally, the g(x)=[(m-1+1)(2n)]=θ(2nm)=θ(nm).

Thus, the time complexity of Accuracy_CAPTCHA_Entry algorithm is θ(nm).

For 100 number of test conducted (m = 100) using 6 various number of CAPTCHAs characters (n = 6), the accuracy has reached 97%. The Table 2 shows that the proposed method has achieved good results rather than other approaches.

Our aim is to maximize the correct recognition for the actual user and zero value recognition for the hackers. The test results are shown in a Table 2 which predicts the usefulness of the proposed CAPTCHA method. It is very clear that an introduction of color, numerals with character and dynamic random order for generating CAPTCHA which got good accomplishment rate for enabling web security.

The Figure 10 shows the linear relation between the length of a CAPTCHA and user performs time. If we set the time limit to solve a dynamic random order CAPTCHA test, bots have a very high negative chance to predict a CAPTCHA character at any time[15].

Table 2. Variant random order CAPTCHA test performed by the user

CAPTCHA Test			Predict all the CAPTCHA character / Human Recognition Human Recognition		
No of characters used	Attacks	Total samples	Sce-nario-1	Sce-nario-2	Sce-nario-3
6 to 8	Dictionary + OCR	100	97%	96%	96%

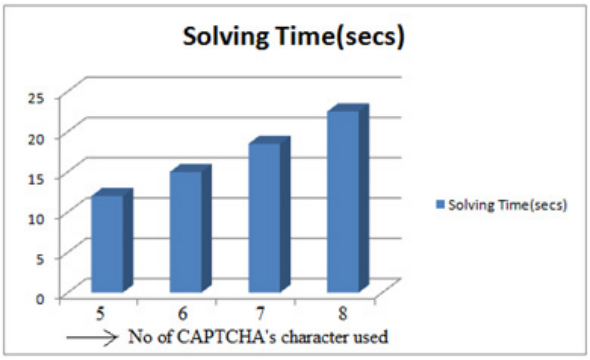

Figure 10. Human user takes time Vs Number of CAPTCHA character used.

6. Conclusion and Future Enhancement

The existing CAPTCHA methods display each character in different colors. These CAPTCHA appears only after the completion of user registration process. But, this paper proposed a new approach for displaying CAPTCHA characters along with a random number generator (RNG). So, the human user can answer the CAPTCHA test based on sorting RNG. The representation of complex CAPTCHA is very easy for the actual user, even though an each and every character may collide with each other and have some crack. At any time, the recognition of CAPTCHA characters is not possible for the bots program. While the user performs the CAPTCHAs test, he/she may not be irritated at any point of time. So, the representation of CAPTCHA is very useful for enhancing the user registration process in any website.

In order to create the fixed time period CAPTCHA in a more securable way, the CAPTCHA characters are to be displayed in random order. If the user wants to write the

CAPTCHA test in the sorted order display with the help of the random numbers, the generated character text will be very useful for enabling registration process.

7. References

1. Chellapilla K, Simard PY. Using Machine Learning to Break Visual Human Interaction Proofs (HIPs). Advances in Neural Information Processing Systems. 2004; 17:265–72.
2. Ur Maulana Azad R. Survey on CAPTCHA systems. Journal of Global Research in Computer Science. 2012; 3(6).
3. Chowdhary NK, Patil R. Captcha's based on the principle – hard to separate text from background. International Journal of Computer Science and Information Technologies. 2014; 5(6):7501–3.
4. Moy G, Jones N, Harkless C, Potter R. Distortion Estimation Techniques in Solving Visual CAPTCHAs. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04). 2004; 2. p. 1–6.
5. Saini BS, Bala A. A review of bot protection using CAPTCHA for web security. IOSR Journal of Computer Engineering (IOSR-JCE). 2013; 8(6):36–42.
6. Von Ahn L, Blum M, Langford J. Telling Human and Computers Apart Automatically. Communications of the ACM. 2004; 47(2):1–11.
7. Baird HS, Loprsti DP. T Converse CAPTCHA generation as a web Service. Proceedings 2nd International Conference on Human Interative proofs. HIP'05. 2005. p. 82–96.
8. Bursztein E, Martin M, Mitchell JC. Text based CAPTCHA Strengths and Weaknesses. ACM Computer and Communication security (CCS). 2011; 1–14.
9. Chandavale AA, Sapkal AM, Jalnekar RM. A Framework to analyze the security of text based CAPTCHA. International Journal of Computer Applications. 2010; 1(27):127–32.
10. Zhu BB, Yan J, Bao G, Yang M, Xu N. Captcha as Graphical Passwords—A New Security Primitive Based on Hard AI Problems. IEEE Transactions on Information Forensics and Security. 2014; 9(6):891–904.
11. Zachariah UE, Swarma Priya RM, Dharun VS. Online Guessing Attacks – A prevention using PGRP and PCCP. Indian Journal of Science and Technology. 2015; 8(26):1–4.
12. Yan J, El Ahmad AS. Breaking Visual CAPTCHAs with Naïve Pattern Recognition Algorithms. 23rd Annual Computer Security Applications Conference, ASPC'07, Miami Beach, FL. 2007. p. 279–91.
13. Kumar V, Yadava PS. Position based Captcha: Changing place restriction mimimize the automatic access. International Journal of Advanced Research in Computer Science and Software Engineering Research. 2013; 3(10):466–71.
14. Shireesha M, Gaikwad Terna VB. Performance Evaluation of CAPTCHA WORD RANKING algorithm to break video CAPTCHA. International Journal of Computer Application. 2013; 75(10):1.
15. Yadava PS, Sahu CP, Shukla SK. Time-variant captcha generating strong captcha by reducing time to automated computer. Journal of Emerging Trend in Computing and Information Science. 2011; 2(12):701–4.